Randomizing Packet Memory Networks for Low-latency Processor-memory Communication

Daichi Fujiki[1], Hiroki Matsutani[1], Michihiro Koibuchi[2], and Hideharu Amano[1]

[1]Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Japan
blackbus@am.ics.keio.ac.jp

[2]National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
koibuchi@nii.ac.jp

Abstract— Three-dimensional stacked memory is considered to be one of the innovative elements for the next-generation computing system, for it provides high bandwidth and energy efficiency. Particularly, packet routing ability of Hybrid Memory Cubes (HMCs) enables new interconnects for the memories, giving flexibility to its topological design space. Since memory-processor communication is latency-sensitive, our challenge is to alleviate latency of the memory interconnection network, which is subject to high overheads from hop-count increase. Interestingly, random network topologies are known to have remarkably low diameter that is even comparable to theoretical Moore graph. In this context, we first propose to exploit the random topologies for the memory networks. Second, we also propose several optimizations to leverage the random topologies to be further adaptive to the latency-sensitive memory-processor communication: communication path length based selection, deterministic minimal routing, and page-size granularity memory mapping. Finally, we present interesting results of our evaluation: the random networks with universal memory access outperformed non-random networks of which memory access was optimally localized.

I. Introduction

Memory wall problem has become a central issue in the computer systems with the growing gaps in the performance between memories and processors. The limitation on the memory bandwidth is imposed from both processor I/O bandwidth and DRAM bandwidth. Recently, three-dimensional stacked memories have been proposed and they are likely to become an important component in the next-generation computer systems. A striking feature of the stacked memories is their performance in channel bandwidth and energy efficiency.

Hybrid Memory Cubes (HMCs) [1], in particular, support packet routing function at the logic die placed at the bottom stack, by the functionality of a switch that can forward the message to another HMC through a path-through link. *Memory network* is recently proposed new paradigm for the memory system; HMCs are viewed as routers, and augmented flexibility in the topological design space enables new interconnection among memories [2], [3]. The memory network benefits to relieve bottlenecks of memory access thanks to the high-speed channel links and enhanced throughput. However, mainly because the links would be subject to dominantly high power consumption, its router is constrained to have small degrees, i.e. four according to HMC specification 2.0[1]. The low-degree HMC routers aggravate the network latency due to increased hop-counts and measurable per-hop temporal cost of them. Clearly, building low-latency memory networks is an emergency topic for extensive use of HMCs. Fortunately, since HMCs can support routing tables for arbitrary routing as suggested in [1] and the power required is not proportional to the link length, the additional overhead to support arbitrary topologies and routings would not impact on area and power. However, the limited number of channels in the memory node settles thorny issue to the scalability of the network, which makes it challenging for typical classic topologies to expand the memory network to the larger scale. These facts stirred up our topology-design interests mainly in favor of better parallel-application performance.

This paper proposes a new approach to alleviate the network latency in the memory network, by introducing random topology based on small-world property. The random networks are attracting interest due to their potential to significantly reduce network diameter and average shortest path length. We investigated further optimization of the random network to best satisfy the requirement for the latency-sensitive memory network. In particular, we selected the random topology based on *communication path length criteria*, and we extended an optimized deterministic deadlock-free routing algorithm [4] to realize *minimal routing* for the selected random topology. Moreover, we exploited *page-size granularity memory mapping* across HMCs for better packet distribution and load balancing. These optimizations enable the memory network to be more scalable even with small number of available channels, while low in the communication latency. Under the condition that only four channels are available in one memory node, the random memory network significantly reduces latency and becomes highly scalable. Our investigations further reveal that the randomized memory network offers better energy efficiency by simply cutting off power of unused links.

The rest of the papers is organized as follows. Background and related work are discussed in Section II. In Section III, we discuss our proposal in terms of network design, topology generation, and our optimization for random memory network. Section IV shows the result of our evaluation through graph analysis, network simulation, and full-system simulation. Fi-

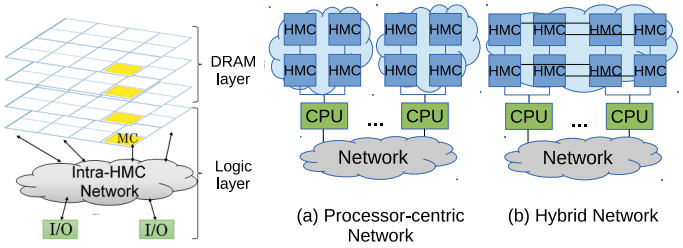

Fig. 1. HMC's block diagram and memory network system designs using HMCs [2].

nally, Section V concludes the paper with a brief summary of our findings.

II. Background and Related Work

A. Hybrid Memory Cubes

Hybrid Memory Cubes (HMCs) [1] are recently proposed 3D stacked memory device which provides high-bandwidth and cost-efficient transactions with DRAM memories [5]. The block diagram of an HMC is shown in Fig. 1. HMCs have several stacks of DRAM layers on top of a logic layer placed at the bottom, and each layer is connected via TSV (Though Silicon Via). Each DRAM layer is partitioned into several segments, and a group of segments in a vertical column of the HMC is called *vault*. The logic layer is composed of several elements: the memory controllers or *vault* controllers which manipulate DRAM access to their vault, I/Os for packet transaction, and a switch which connect the memory controllers and the I/Os.

The communication between CPU-HMC and HMC-HMC is done through high-speed signaling, which contributes to broaden channel bandwidth. On top of that, they use high-level messages for communication, such as request/response messages. This allows processors to communicate with abstracted memory objects without any consideration of detailed DRAM technologies and unique timing of DRAM, and those are now managed by an HMC's memory controller instead. Notably, HMCs can be chained to increase the memory capacity. This enables arbitrary deterministic routing and topology with a routing table in which each entry corresponds to a destination HMC identifier (CubeID).

B. Memory Network

Since it is not only I/O–memory controller traversals but I/O–I/O traversals that are supported by the switch at the logic layer of HMCs, they can forward packets to other HMCs. Kim et al. [2], [3] proposed a new system design using HMCs in order to fully utilize the bandwidth of the high-speed channel links across all system components, by looking HMCs as routers of a network. They defined conventional system design, which assigns processor channels to dedicated memory channels and dedicated processor interconnection channels, as (conservative) processor-centric network (PCN) (Fig. 1(a)); and then proposed memory-centric network (MCN) as the system design with more flexible bandwidth adaptation by connecting all processor channels to HMCs. They also proposed hybrid network (HBN) (Fig. 1(b)) as the combination of PCN and MCN.

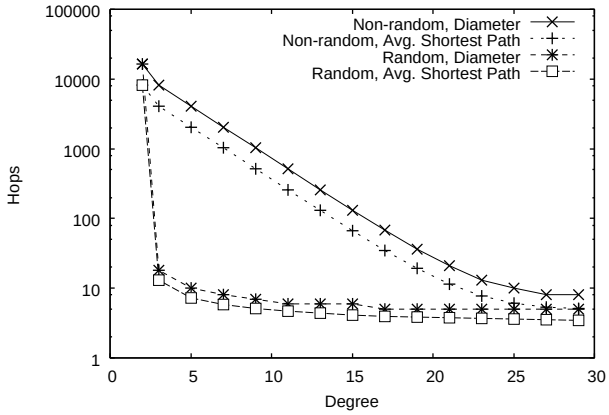

Fig. 2. Diameter and average shortest path length vs. degree for a ring topology with non-random or random shortcuts, for 2^{15} vertices [7].

Our analysis of memory network traffic observed much more inter-processor communication packets traversing in the network. Thus, it would be a natural solution to keep the processor-to-processor communication channels in the memory network. Indeed, most of the ongoing processor architecture clearly specifies distinct memory access interfaces and processor interconnection interfaces. For example, SPARC64 XIfx [6], Fujitsu's processor designed for HPC systems, has I/Os composed of Tofu2 interface for inter-processor communication and 128-lane HMC interface for memory access. These observation made us presumed that HBN is the feasible design for both processor vendors and memory vendors, giving consideration to current processor organizations. This study thus sets HBN organization as the baseline.

C. Random Shortcut Topology

Small world property is known to drastically reduce network diameter by introducing random links. Various researches are focusing on this property to apply it to different fields, such as computer data network and wireless sensor network. We explored the availability of the random shortcut topology for high performance computing (HPC) systems in [7]. We found that ring topology with random shortcut links effectively reduced network diameter and average shortest-path length, and we suggested that randomized topology could organize low-latency, high-throughput HPC network. The similar effect occurs in network-on-chip [8], [9].

Fig. 2 from our study [7] elaborates on an example of the striking features of randomized topologies. We added shortcut links randomly or non-randomly to a ring topology. The non-random topologies were created by adding K links to each vertex i of N-vertex ring to connect vertices $i + \lfloor N/2^k \rfloor \bmod N$, for $k = 1, \cdots, K$. Thus, the diameter was reduced by approximately a factor of two. On the contrary, random topologies, each vertex of which was added randomly directed shortcut links of the same number, observed significant reduction in diameter and average shortest path

length, both of which resulted in dramatically low values compared with non-random. Moreover, these improvements required only a small number of random links as can be seen in Fig. 2.

Therefore, we presume that latency-sensitive memory network, which can afford only a few number of links per node and is prone to deteriorate its performance directly by the increase in hop-count, should benefit from reduction in average shortest-path length by organizing randomized topology.

III. Network Design

This section describes our baseline network design and the organization of randomized memory network.

A. Links and Processor Channels

HMC Specification 2.0 [1] stipulates that an HMC has four full-duplex bidirectional links, on which serialized messages traverse for communication. The system assigns unique ID in the network called CubeID to each HMC, and the routing functionality of each HMC refers to the CubeID field in the header of incoming request message for routing computation. Since this ID is a 3-bit value, eight HMCs are available in a system according to the specification. However, in consideration of growing demands for large scale high-performance memory system such as those used in next-generation warehouse-scale computers [10] and large in-memory database, more and more HMCs are supposed to be implemented in a system. Therefore, extending the bit width to distinguish more HMCs is generally accepted [2]. Therefore, this paper also assumes that more than eight HMCs are available in a system.

On the contrary, installing more channel links to an HMC would be challenging because of high energy consumption of each link, which is considered to dominate total power consumption of HMCs (73% on average according to [11]), as well as the need for additional logic modules such as SerDes. This can be revealed from the fact that the number of links in an HMC was decreased when the HMC specification revised from version 1.1 to 2.0. Therefore, we assume low-radix network of which degree is four according to the latest HMC specification, so that such network devises of limited number of links can organize scalable memory network by our methodology.

The number of memory channels gives direct impacts to the system performance. The memory channels denotes the host links that connect a processor with HMCs, and no doubt that the increase in the number of memory channels broadens the memory access bandwidth. Unfortunately, at present, the number of memory channels is severely limited. For example, SPARC64 XIfx [6] has 128-lane HMC interface for memory access on HPC systems. If an HMC is connected with full-width link, a memory controller on SPARK64 XIfx can support at most four direct connections to HMCs. This paper examines with different number of memory channels so as to indicate that various processors with different channel organizations could benefit from randomization.

These 4-degree limitations allow a few conventional competitor topologies. Interconnection networks can be classified into two types: direct and indirect, and among the direct interconnection networks few topologies are known to be viable under the degree of four or below, e.g. mesh, tori, and spidergon. We compared our random topology (Fig. 6) with counterpart single/multi-channel mesh (Fig. 3) and De Bruijn network (Fig. 5), and single-channel dragonfly [12] (Fig. 4).

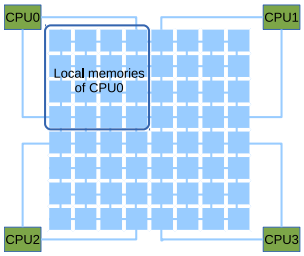

Fig. 3. Memory channel connection of mesh with minimal local HMC diameter ($n_{ch} = 2$).

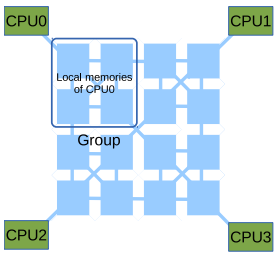

Fig. 4. Dragonfly [12].

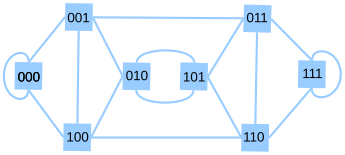

Fig. 5. De Bruijn graph of order 3.

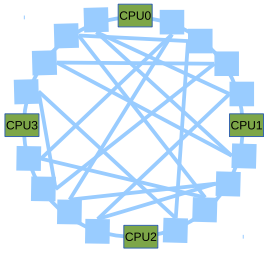

Fig. 6. Schematic of a randomized memory network.

B. Topology and Memory Access Design

1) UMA and NUMA in Memory Network: The memory network offers a choice of two types of memory access design: UMA (Uniform Memory Access) and NUMA (Non-Uniform Memory Access). UMA uses shared address space across all HMCs, while here NUMA defines distinct local address spaces for each processor with local HMC groups. UMA seems to be meaningful structure for the memory network as a processor can access to any HMCs across the network. However, given that HMCs in the mesh network accessed uniformly from a processor, inevitable increase in hop-count to access distant HMCs generates additional latency. Conversely, although remote processors are required to relay remote HMC access packets in NUMA, localized HMC arrangement in the vicinity of home processors reduces network diameter for local packets, resulting in low-latency. For example, assuming 8 $\times$ 8 mesh HBN with uniform traffic, average shortest-path length (ASPL) from a processor is 8.0 for UMA, while just 4.75 for NUMA. Therefore, we employ NUMA with directory-based MOESI cache coherence protocol.

On the other hand, in a random topology, local HMCs and remote HMCs are placed randomly even for NUMA. Thus, the memory traffic traverses all across the network regardless of UMA and NUMA. We assumed both random and conventional topology use NUMA for fair comparison; however, we leave

Algorithm 1 Random topology generation

$N \leftarrow$ Set of nodes (with free ports) in a given system
$fp(n) \leftarrow$ Number of free ports of node n
while $N \neq \emptyset$ **do**
    $a \leftarrow$ A node poped from N
    $A \leftarrow$ Set of connected nodes with a
    while $fp(a) > 0$ **do**
        if $N - A = \emptyset$ **then** initialize again
        Randomly choose b from $N - A$
        Connect a with b
        if $fp(b) = 0$ **then** remove b from N
    end while
end while
Check if all nodes are connected
for all processor nodes **do**
    Calculate and add up the shortest path length to each memory node (communication paths)
end for
Calculate ASCPL from the shortest communication path length sum of each processor node

it our future work to explore better memory access and cache coherence protocol for random topologies.

2) Topological Design for Mesh: As mentioned in the previous section, local HMCs in a mesh network are placed near their home processor for the efficient NUMA memory access. We assumed that HMCs had four degrees, thus interfaces to the processors are offered by peripheral HMCs. This work employs XY deterministic routing for mesh, hence we chose the best interface ports so that the ASPL in a local HMC group becomes minimal, as can be seen in Fig. 3.

C. Random Network Optimization

1) Topology Selection Based on Communication Path Length: What count is to consider the difference of the traffic patterns of memory networks from those of other systems such as data-center networks and HPC systems. Generally, random topology with minimum ASPL is preferable to be beneficial for the traffic injected from any nodes. However, all traffic in memory networks is CPU vs. HMC, and no end-to-end communication is observed between HMCs. Therefore, we designed our random topology focusing on the communication paths between CPUs and HMCs, so as to minimize them.

In particular, we selected the topology of the minimum average shortest communication path length (ASCPL) among 100,000 trial of generation of Alrogithm 1. Note that the computation cost of Alrogithm 1 does not a matter because costly calculation of ASPL for all node-combinations is unnecessary and the procedure can be easily parallelized, e.g. it spent only 4.5 hours for extremely large networks, 4-degree 1,024 nodes, with OpenMP API computed on Xeon E5-2470.

2) Deadlock-free Minimal Routing: Among various routing strategies available for irregular networks, up*/down* routing is one of the basic and useful routing schemes. It realizes deadlock-free non-minimal routing by using hierarchical structure of a spanning tree of a certain root node. However, non-minimal routing is not preferable for the memory networks of which latency and performance easily degrade by the increase

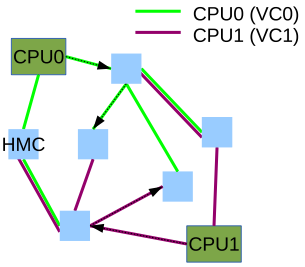

Fig. 7. Minimal routing for random memory networks.

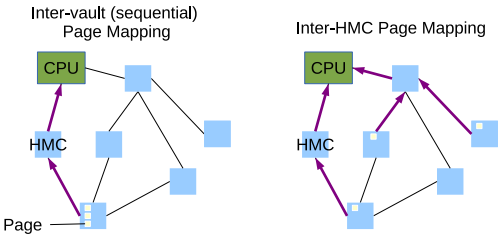

Fig. 8. Memory mapping policies.

of hop-count. For this reason, we extended the up*/down* routing to offer deadlock-free deterministic *minimal* routing optimized to irregular memory network to further reduce latency. As mentioned in section III-C1, memory networks only observe CPU-involved traffic. We focus on this characteristic, and route packets on the spanning trees of which roots are set at the processor node. This can be regarded as an extended version of our work [4] and we optimized it for the memory networks. In particular, following procedures are taken:

1) For each processors, assign VCs (Virtual Channels) and search for a spanning tree, setting the processors as the root node (root processor).
2) Each memory node creates routing tables using the tree structures just as those of up*/down* routing for each VCs.
3) Messages of which source node or destination node is the root processor exclusively use the processor's VCs, and VC switching to other processors' VCs does not occur in the network.

Fig. 7 shows the concept of this routing scheme. In this way, round-trip message transfer toward a destination HMC only uses the down channels (CPU-to-HMC) or up channels (HMC-to-CPU) of the root processor's VCs. The minimal paths are always taken because the processor node is the root of the spanning tree. Moreover, deadlock freedom is obvious because of no cyclic dependency in each VC and no possibility of switching to VCs of the other processor's domain.

3) Inter-HMC Page-size Granularity Memory Mapping: Traffic congestion could happen in the memory networks, for some workloads has relatively high access locality. We attempted to contend this problem by the minimal routing and *inter-HMC* page-size granularity memory mapping, taking advantage of the randomized network. We assumed that each HMC has 16 memory controllers, and we applied round-robin

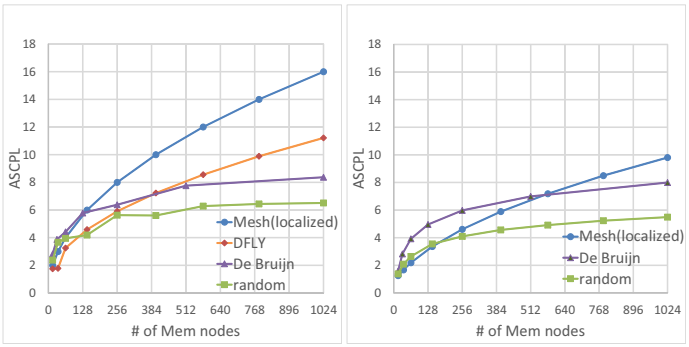

Fig. 9. Changes in ASCPL in a system with $n_{ch} = 1$ (left), 4 (right).

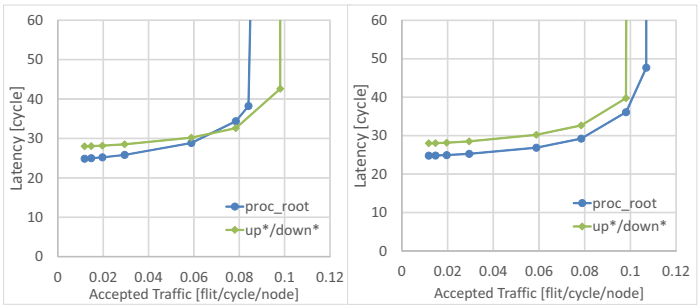

Fig. 10. Routing comparison (left: 4VC, right: 8VC).

page-size memory mapping policy *across* memory controllers of *different* HMCs. In particular, page address a is mapped to a memory controller $m_h (0 \leq m_h \leq 15)$ in HMC $h (0 \leq h \leq n_{L_p} - 1)$ according to the following equation:

$$h = \mathtt{a} \bmod n_{L_p}, m_h = \mathtt{a}/n_{L_p} \bmod 16,$$

where n_{L_p} denotes the number of local HMCs of processor p. In this way, memory interleaving across the network would ease traffic congestion and accomplish load balancing. This is because HMC connection is randomized in the network, unlike typical regular topologies where methodically arranged HMCs would cause congestion with adjacent HMCs. Fig. 8 shows the conceptual diagram in comparison with the conventional inter-vaults memory mapping policy.

In addition, random network would drastically reduce network diameter and ASCPL, as well as deterministic routing simplifies the routing fabrics and flow control. Our analysis shows that in 1,024 node network the ASCPL of the random topology is approximately 55.9% of that of mesh as revealed in the following section. Thus, increase in latency would be moderate even when the network size becomes large.

IV. Evaluation

This section describes the methodology and result of our evaluation for randomized memory network. In particular, we analyzed the effect of our random topology by graph analysis, and then we evaluated our proposed routing scheme by a cycle-accurate network simulator with synthetic traffic. Finally, we used a full-system simulator to evaluate whole random memory network system with real workloads from the viewpoint of performance and energy consumption.

A. Hop-count Analysis of Memory-processor Communications

In the first set of analyses, we assumed uniform traffic traversing the network, and calculated average shortest communication path length (ASCPL). For mesh and dragonfly network, we assumed that the processors only accessed their local memories, and calculated ASCPL from following equation: $\sum_{p \in P} \sum_{l \in L_p} d_{p,l}/n$, where P is the set of processor nodes, n is the number of total memory nodes, L_p is the set of the local memory nodes of processor p, and $d_{a,b}$ is the shortest path length from node a to node b. Conversely, for De Bruijn and random network, we assumed universal traffic without distinction of local nodes, and calculated ASCPL as follows: $\sum_{p \in P} \sum_{m \in M} d_{p,m}/n$, where M is the set of all memory nodes.

We assumed that the system had four processors, and we analyzed under three conditions of the number of processors' memory channels n_{ch}: $n_{ch} = 1, 2, 4, 8$. Also, due to the limitation of the number of HMC links, we evaluated mesh, dragonfly (DFLY), De Bruijn and random under the condition of $n_{ch} = 1$; mesh, De Bruijn and random under $n_{ch} > 1$.

These regular topologies, i.e. mesh, dragonfly and De Bruijn, are justifiable as the comparison objectives, for HMCs share the same condition that degree = 4. De Bruijn, which contains long range links, is selected as a competitor because it has great advantage in diameter, i.e. it affords the diameter of hypercube while the degree is the same as 2D mesh. However, in terms of the memory network, mesh in particular is advantageous in a sense that it can be optimally localized as mentioned in section III-B2, while random and De Bruijn face difficulty for it. Note that the dragonfly can be applied only when $n_{ch} = 1$ since only HMCs placed at the corners have an available port to connect to CPU. Some research investigates other topologies such as flatten butterfly [2], however, it omits the 4-degree condition of up-to-dated HMC specification. Likewise, 2D tori have trouble because no HMC affords a port for CPU.

Fig. 9 shows the results of our graph analysis. The vertical axis explains ASCPL, while the horizontal axis indicates the number of total memory nodes. While not in the figure, the almost same tendency as $n_{ch} = 4$ was observed when $n_{ch} = 2, 8$. The result demonstrated that small scale networks with less than 150 nodes increased the ASCPL of random; however, the notable result to emerge from the data was that large scale networks with more than 180 nodes found random topology had smaller ASCPL than any other regular topologies despite the assumption of universal access to all nodes. Although the random yielded worse ASCPL in smaller scale network where the counterparts' access locality predominated, we can also validate the randomization, for it can provide universal memory access with relatively low overhead.An important observation of these findings is that the random network efficiently reduces ASCPL for the large scale memory networks, and that it becomes more beneficial as the network size grows.

TABLE I
SYSTEM CONFIGURATION

Parameter	Configuration
Core	Out-of-Order core @ 3.5GHz
L1 I/D cache	32KB, 2-way
L2 cache	Private 256KB, 8-way
Cache coherence	Directory-based MOESI
Cacheline size	64B
HMC Configuration	8 layers, 16 vaults, 4 I/O
Total memory	8GB
DRAM Configuration	tCK=0.8ns, tRP=13.75, tRCD=13.75, tCL=13.75, tWR=15, tRAS=27.5

B. Network Simulation

This evaluation explores the impact of our deadlock-free minimal deterministic routing scheme for irregular memory network by a cycle-accurate network simulator [13]. Here we assumed 4CPU-64HMC random memory network system. Random synthetic traffic of CPU-to-HMC and HMC-to-CPU are injected to the network, and we assessed difference of latency between up*/down* and our scheme. The root node for the up*/down* routing was properly chosen so that the difference in average communication path lengths from each processor was minimized to make it fair. Likewise, for fair comparison, same number of VCs were set to be available for both routing schemes, and up*/down* routing was allowed to choose any VCs while routing.

Fig. 10 shows the result of the network simulations. It indicates that proposed routing scheme certainly reduced network latency. When four VCs were used, the proposed minimal routing decreased the latency by 11.3%, and the average hop-count by 14.1%. Because the result is of *minimal* routing, this is the minimum average hop-count value theoretically possible.

In Fig. 10 (left), proposed routing observed little decrease in the throughput. This should be attributed to the bottleneck of available VCs in a spanning tree: proposed routing could use only one VC if total number of VCs was four, while up*/down* could choose a free VC among all. This bottleneck can be avoided by augmenting the number of VCs. Fig. 10 (right) is the result of eight VC configuration providing two for each spanning tree, and it shows that proposed routing offered 109% throughput than up*/down* while low in the latency.

C. Full-system Simulation

To evaluate performance of the random memory network for real-workloads, we used Gem5 simulator [14]. Ruby system, gem5's memory subsystem models, was tailored for use with modeled HMCs and the page-size granularity memory mapping policy. Note that the inter-HMC page-size mapping was only applied to random topologies, because regular topologies suffered from adjacent congestion. Likewise, gem5's detailed interconnection network model Garnet was customized to support random network and proposed routing schemes. The parameters for the full-system simulation are listed in Table I. We used NAS Parallel Benchmarks for the evaluation.

This paper assumes to use deterministic routing in consideration of simplifying flow controls and lowering latency. In the first set of evaluations, we assessed the system performance of MCN-Mesh and MCN-random, which did not have P2P CPU interconnect; HB-Mesh, HB-DFLY, and HB-random, which had P2P interconnect; to explore the system design that best suit for the deterministic routing scheme, assuming 4CPU-64HMC system. The result is shown in Fig. 11.

Fig. 11 shows that MCNs suffered from hop-count overhead for inter-CPU communication, resulting in degradation in the performance. Since our analysis revealed remote CPU access including remote memory access counted several hundred times of that of local-memory access, we consider retaining inter-processor direct links to be important to reduce latency. Thus, we suppose HBN for further analyses.

From our results of HBN, we found that the random network demonstrates equivalent or slightly better performance to the regular networks when $n_{ch} = 1$. As substantiated in Fig. 9, gaps in ASCPL grow as the network size increases, thus we believe the performance of random network would not be reversed in larger networks. On the contrary, mesh outperformed by at most 6.1% with multiple memory channel as anticipated in the graph analysis. Hereafter, we go further to explore the impact of the increase in network size for each topologies.

Fig. 12 and 13 show the changes in cycles for mesh, De Bruijn and random topology under the condition of $n_{ch} = 1$ and 4, setting the network size to 16, 64, 256, and 1,024. The figures are normalized to those of mesh of each network size. While the random network shows comparable performance to that of regular topologies when the network size is small (16, 64), the remarkable result to emerge from the data is that the random outperforms for all the applications when the network size becomes large. Specifically, for 1,024 node network, the random reduced cycles by 21.1% at maximum (EP), 6.6% on average (GMEAN) when $n_{ch} = 4$ compared to mesh. De Bruijn shows comparable performance to the random; however, few results are reported to substantiate the performance superiority of De Bruijn to random.

Fig. 14 through Fig. 16 shows the cycle count overhead, which increases as the network size grows. The values are normalized to those of network size 16, and they are correspondent to Fig. 13. It was found that mesh network increased cycle overheads steeply due to significant increase in hop-count to destination nodes even if accessible nodes were limited to those of local memories. On the other hand, it is remarkable that random showed moderate increase in cycle overhead because ASCPL growth was also moderate in accordance with the graph analysis, even compared to De Bruijn. These results show the satisfactory agreement between graph analysis and full-system simulation.

D. Energy Evaluation

We conducted an analysis of the amount of energy consumed by the network in real application workloads we used in the previous section. We used an interconnect energy model of the memory networks described in [2]. We assumed 2.0 pJ/bit for Tx&Rx real packets, 1.5 pJ/bit for idle packets and 1.0 pJ/bit for processing of a router according to their

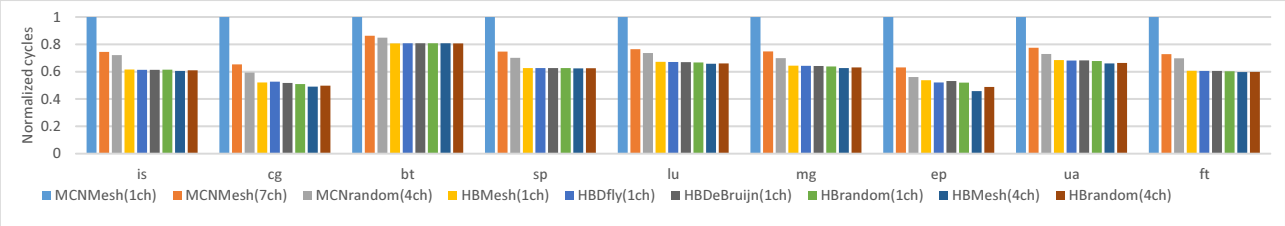

Fig. 11. Performance comparison of different network design (4CPU-64HMC).

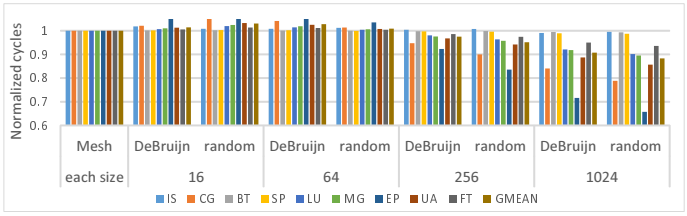

Fig. 12. Performance comparison of different network size ($n_{ch} = 1$).

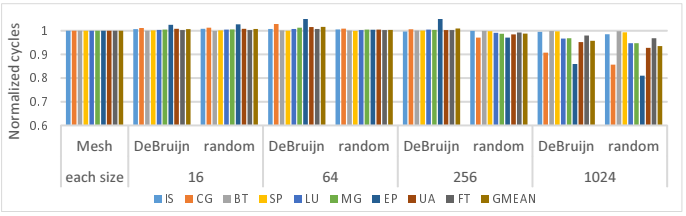

Fig. 13. Performance comparison of different network size ($n_{ch} = 4$).

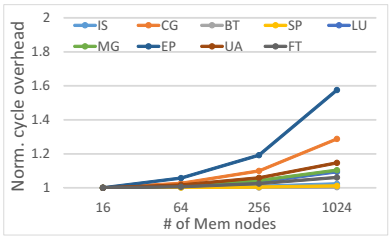

Fig. 14. Cycle overhead ($n_{ch} = 4$, Mesh).

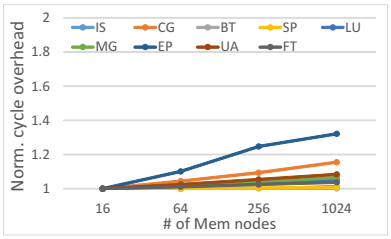

Fig. 15. Cycle overhead ($n_{ch} = 4$, De Bruijn).

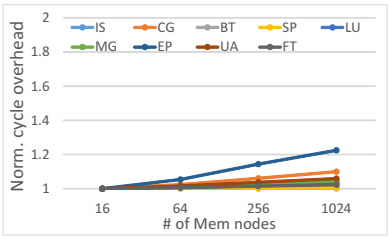

Fig. 16. Cycle overhead ($n_{ch} = 4$, random).

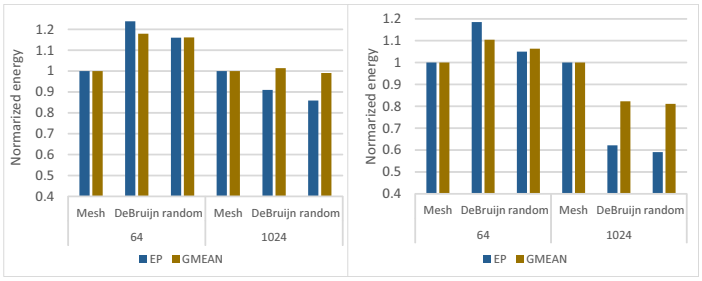

Fig. 17. Energy consumption (left: all links activated, right: power-gated).

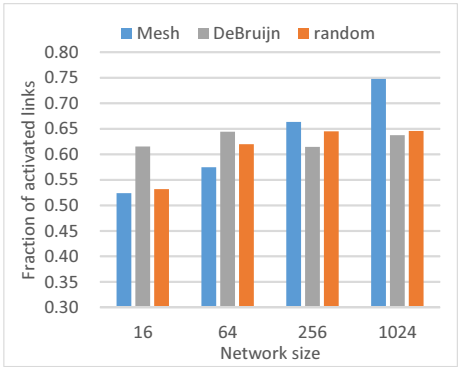

Fig. 18. The fraction of activated links.

work.

Presumably, the random and De Bruijn of which links outnumber the mesh would consume more power, since idle packets need to be exchanged over the high-speed signaling links when there is no communication data. Specifically, some HMCs in mesh network, such as those at periphery, do not fully use their available links while those in random use all, and at first this balance seemed to lead the scenario that random was the topology of high-performance, and of high-energy consumption to make a trade-off.

When a memory network employs a minimal routing, some of the HMC-to-HMC links are not utilized. These links can be power-gated, and we thus assumed no power was consumed for them. Fig. 17 shows the difference in total energy consumption with and without the constant power-gating. EP is chosen as the metrics since it showed most sensitive reaction. When the network size is small, the energy consumption for idle links is the major detriment of energy efficiency of random and De Bruijn, and thus it becomes rather comparable to mesh when the constant power-gating is implemented. Interestingly, this feature becomes more prominent for the large networks, where 35% of total links are power-gated in random memory network while just 25% in mesh when the network size is 1,024. Fig. 18 shows the fraction of activated links among total available links, and this demonstrates that the amount of activated links is on the same level with mesh in small size network, still growing not as high as mesh even if the networks

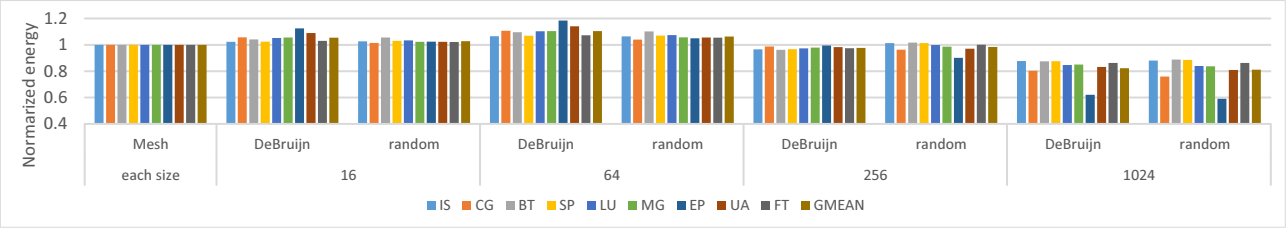

Fig. 19. Energy comparison of different network size ($n_{ch} = 4$, power-gated).

become large.

Contrary to our initial expectation, the results of energy evaluation as shown in Fig. 19 remarkably reveals that the energy consumption of random is highly comparable to mesh, and it becomes rather advantageous when the network size becomes large. This is explained by two factors: the number of activated links became smaller as just stated above, and the total cycles decreased as demonstrated in the performance evaluation. The difference in random and De Bruijn can be explained by almost the same factors, while the former is rather prominent in small networks and the latter in large, as in Fig. 18 and 13. Finally, for 1,024 node network, the random reduced energy by 40.9% at maximum (EP), 18.9% on average (GMEAN) when $n_{ch} = 4$ compared to mesh.

Summing up the results, our works led us to conclude that our random network is the scalable, high-performance and energy-efficient organization for large-scale memory network.

V. Conclusions

This paper proposed to introduce random networks based on small-world property to organize scalable and low-latency memory network. We also proposed communication path length based selection for random graph generation, minimal deterministic routing scheme, and inter-HMC page-size granularity address mapping, to effectively reduce latency even in a large network. In the memory network, routing and link length do not heavily impact on both the area and power overhead of HMCs, because of the use of routing tables in HMCs and the irrelevancy of link power consumption to its length. Our interests thus focus on the performance gain of parallel applications.

Our main concern about the randomized memory network was that increased utilization rate of HMC links should lead an upsurge in energy consumption, but conversely, adequately powered off links save needless idle energy, resulting in better energy performance, which is around 18.9% less than optimized mesh on average. The major trade off of the randomized memory network would thus lie in the complexity of actual layout on printed circuit boards. Despite this we believe our work could be a springboard for large scale memory network using randomized topology.

In conclusion, our proposed method reduced cycles by 6.6% on average, according to the result of our full-system simulation. Interestingly, those are the evidence that random network outperformed the counterpart mesh network, which is optimized by limiting the access range to processors' local HMCs. Furthermore, predominance over such regular topologies with long range links as De Bruijn is also proven. Consequently, random topology would enable further optimization for parallel applications due to increased accessibility to remote memory nodes with reduced latency.

Acknowledgment

A part of this work was supported by SCOPE.

References

[1] H. M. C. Consortium, "Hybrid Memory Cube Specification 2 . 0 Hybrid Memory Cube with HMC-30G-VSR PHY," 2014. [Online]. Available: http://www.hybridmemorycube.org/

[2] G. Kim, J. Kim, J. H. Ahn, and J. Kim, "Memory-centric system interconnect design with hybrid memory cubes," in *Parallel Architectures and Compilation Techniques (PACT), 2013 22nd International Conference on*, Sept 2013, pp. 145–155.

[3] G. Kim, M. Lee, J. Jeong, and J. Kim, "Multi-gpu system design with memory networks," in *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, Dec 2014, pp. 484–495.

[4] H. Matsutani, P. Bogdan, R. Marculescu, Y. Take, D. Sasaki, H. Zhang, M. Koibuchi, T. Kuroda, and H. Amano, "A case for wireless 3d nocs for cmps," in *18th Asia and South Pacific Design Automation Conference (ASP-DAC), 2013, Yokohama, Japan, January 22-25, 2013*, 2013, pp. 23–28.

[5] J. Pawlowski, "Hybrid memory cube (HMC)," *Hotchips*, pp. 1–24, 2011.

[6] T. Yoshida, "SPARC64 XIfx : Fujitsu's next generation processor for HPC," *Hotchips*, pp. 1–31 (in slides), 2014.

[7] M. Koibuchi, H. Matsutani, H. Amano, D. Hsu, and H. Casanova, "A case for random shortcut topologies for hpc interconnects," in *Computer Architecture (ISCA), 2012 39th Annual International Symposium on*, June 2012, pp. 177–188.

[8] H. Matsutani, M. Koibuchi, I. Fujiwara, T. Kagami, Y. Take, T. Kuroda, P. Bogdan, R. Marculescu, and H. Amano, "Low-latency wireless 3D NoCs via randomized shortcut chips," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*, 2014, pp. 1–6.

[9] H. Yang, J. Tripathi, N. Enright Jerger, and D. Gibson, "Dodec: Random-link, low-radix on-chip networks," in *Proceedings of the International Symposium on Microarchitecture*, 2014.

[10] K. Asanović and D. Paerson, "FireBox : Upcoming Applications," *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST), 2014 - Keynote presentation*, 2014. [Online]. Available: https://www.usenix.org/conference/fast14/technical-sessions/presentation/keynote

[11] J. Ahn, S. Yoo, and K. Choi, "Dynamic power management of off-chip links for hybrid memory cubes," in *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, June 2014, pp. 1–6.

[12] J. Kim, W. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Computer Architecture (ISCA), 2008 35th International Symposium on*, June 2008, pp. 77–88.

[13] N. Jiang, D. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. Shaw, J. Kim, and W. Dally, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, April 2013, pp. 86–96.

[14] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.